An implicit multishift QR-algorithm for Hermitian plus low rank matrices

Raf Vandebril and Gianna M. Del Corso

Raf Vandebril
Department of Computer Science
KU Leuven, Belgium
Raf.Vandebril@cs.kuleuven.be

Gianna M. Del Corso
Department of Computer Science
Università di Pisa, Italy
DelCorso@di.unipi.it

Abstract

Hermitian plus possibly non-Hermitian low rank matrices can be efficiently reduced into Hessenberg form. The resulting Hessenberg matrix can still be written as the sum of a Hermitian plus low rank matrix. In this paper we develop a new implicit multishift QR-algorithm for Hessenberg matrices, which are the sum of a Hermitian plus a possibly non-Hermitian low rank correction. The proposed algorithm exploits both the symmetry and low rank structure to obtain a QR-step involving only $\mathcal{O}(n)$ floating point operations instead of the standard $\mathcal{O}(n^2)$ operations needed for performing a QR-step on a Hessenberg matrix. The algorithm is based on a suitable $\mathcal{O}(n)$ representation of the Hessenberg matrix. The low rank parts present in both the Hermitian and low rank part of the sum are compactly stored by a sequence of Givens transformations and a few vectors. Due to the new representation, we cannot apply classical deflation techniques for Hessenberg matrices. A new, efficient technique is developed to overcome this problem. Some numerical experiments based on matrices arising in applications are performed. The experiments illustrate effectiveness and accuracy of both the QR-algorithm and the newly developed deflation technique.

Article information

- Vandebril, Raf; Del Corso, Gianna M.. *An implicit multishift QR-algorithm for Hermitian plus low rank matrices*, SIAM Journal on Scientific Computing, volume 32, issue 4, pages 2190-2212, 2010.
- This article equals the final publisher's version. A link is found below.
- Journal's homepage: https://www.siam.org/journals/sisc.php
- Published version: http://dx.doi.org/10.1137/090754522
- KU Leuven's repository url: https://lirias.kuleuven.be/handle/123456789/280643

KU LEUVEN

www.kuleuven.be

SIAM J. SCI. COMPUT.
Vol. 32, No. 4, pp. 2190–2212
© 2010 Society for Industrial and Applied Mathematics

AN IMPLICIT MULTISHIFT QR-ALGORITHM FOR HERMITIAN PLUS LOW RANK MATRICES*

RAF VANDEBRIL† AND GIANNA M. DEL CORSO‡

Abstract. Hermitian plus possibly non-Hermitian low rank matrices can be efficiently reduced into Hessenberg form. The resulting Hessenberg matrix can still be written as the sum of a Hermitian plus low rank matrix. In this paper we develop a new implicit multishift QR-algorithm for Hessenberg matrices, which are the sum of a Hermitian plus a possibly non-Hermitian low rank correction. The proposed algorithm exploits both the symmetry and low rank structure to obtain a QR-step involving only $\mathcal{O}(n)$ floating point operations instead of the standard $\mathcal{O}(n^2)$ operations needed for performing a QR-step on a Hessenberg matrix. The algorithm is based on a suitable $\mathcal{O}(n)$ representation of the Hessenberg matrix. The low rank parts present in both the Hermitian and low rank part of the sum are compactly stored by a sequence of Givens transformations and a few vectors. Due to the new representation, we cannot apply classical deflation techniques for Hessenberg matrices. A new, efficient technique is developed to overcome this problem. Some numerical experiments based on matrices arising in applications are performed. The experiments illustrate effectiveness and accuracy of both the QR-algorithm and the newly developed deflation technique.

Key words. Hermitian plus low rank matrices, Givens-weight representation, implicit method, multishift, QR-algorithm

AMS subject classification. 65F15

DOI. 10.1137/090754522

1. Introduction.

In the last few years many numerical techniques for computing eigenvalues of structured rank matrices have been proposed (see, e.g., [19, 2, 4] and the references therein). In particular, the QR-algorithm received a great deal of this attention. Suitable representations such as quasiseparable, diagonal-subdiagonal, or Givens-weight to represent the structured rank matrix are essential for the development of effective and efficient algorithms. A representation of the structured rank Hessenberg matrix in terms of $\mathcal{O}(n)$ parameters makes it possible to perform QR-steps in $\mathcal{O}(n)$ operations instead of the $\mathcal{O}(n^2)$ operations required by standard methods.

This paper is concerned with the efficient computation of all the eigenvalues of an $n \times n$ Hermitian matrix perturbed by a possibly non-Hermitian low rank correction. This problem arises in different situations, for example, when a perfectly Hermitian structure is corrupted by errors in only a few rows or columns. Another typical situation is the eigenvalues' computation of matrices with the sizes of the off-diagonal elements decaying rapidly away from the main diagonal. In this case, it can be computationally convenient to approximate, up to a desired accuracy, the original matrix by the sum of a Hermitian and a low rank matrix. In many other cases the problem to be solved is modeled naturally in this form (see the numerical experiments). A typical

*Received by the editors March 31, 2009; accepted for publication (in revised form) May 17, 2010; published electronically July 29, 2010. This research was partially supported by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, Belgian Network DYSCO (Dynamical Systems, Control, and Optimization).

http://www.siam.org/journals/sisc/32-4/75452.html

†Department of Computer Science, KULeuven, 3001 Leuven (Heverlee), Belgium (raf.vandebril@cs.kuleuven.be). This author has a grant as "Postdoctoraal Onderzoeker" from the Fund for Scientific Research–Flanders (Belgium).

‡Department of Computer Science, Università di Pisa, Largo Pontecorvo, 3, 56127 Pisa, Italy (delcorso@di.unipi.it). This author was partially supported by the PRIN project: Analisi di strutture di matrici: Aspetti teorici, computazionali e applicazioni.

example is the computation of roots of a polynomial expressed in a basis satisfying a three-term recurrence giving rise to a Hermitian tridiagonal matrix plus a rank-one correction. These matrices are referred to in the literature as comrade matrices [1].

More precisely, let $A = S + UV^H$, $A \in \mathbb{C}^{n\times n}$, where S is Hermitian and $U, V \in \mathbb{C}^{n\times m}$, with $m \ll n$. A standard approach to computing the eigenvalues of A consists in a preliminary reduction to Hessenberg form followed by the standard QR-algorithm. The reduction to Hessenberg form takes $\mathcal{O}(n^3)$ operations in general, but often one can take advantage of the rank structure present in the original matrix and reduce the cost to $\mathcal{O}(n^2)$ [8, 9, 13]. It can be easily seen that the resulting Hessenberg matrix H can still be expressed as the sum of a Hermitian and a low rank perturbation matrix.

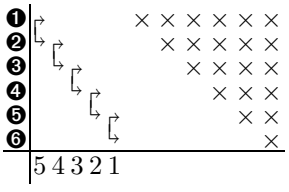
In this paper a new suitable $\mathcal{O}(n)$ representation based on Givens transformations of the Hessenberg matrix is first proposed. This representation allows performing implicit multishift QR-steps involving only $\mathcal{O}(n)$ operations for each iteration. Since the new representation makes deflation in the classical sense impossible, a solution to this problem is also proposed.

In [7] QR-algorithms for Hessenberg matrices which are the sum of symmetric plus low rank matrices were also presented. Our paper differs significantly from theirs. The authors in [7] use the quasiseparable representation; i.e., they store the low rank part by means of few vectors. Instead we use unitary 2×2 transformations for storing the low rank part. Since the two representations are different, the resulting QR-steps and implementations are also completely different. In [7] the standard deflation technique as present in Lapack is used. Using, however, the representation presented here, this is not possible, and both the deflation technique and the successive QR-steps on submatrices need to be adapted. Similar numerical experiments were performed, making it possible to compare the two approaches.

2. Preliminary results. In this section we first discuss preliminary results essential for understanding this article, and, in particular, we present the graphical schemes used for representing the QR-factorization of a Hessenberg matrix. Second, we discuss a similarity transformation of a Hermitian plus low rank matrix to a Hessenberg matrix, which is the sum of a Hermitian plus low rank matrix. It is shown in the next section that the number of parameters for representing this Hessenberg matrix is linear. For developing an implicit QR-algorithm, preservation of the structure is essential. This is discussed in subsection 2.3, where the general scheme of implicit QR-iterations is given.

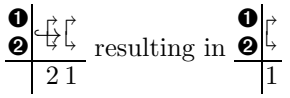
2.1. Working with Givens transformations. Matrices with a rank structure can be represented with an adapted form of the Givens-weight representation [5, 16, 18]. Sequences of Givens transformations admit an easily understandable graphical representation. This representation will be used throughout the paper to show how the algorithm works. For efficient use of this representation, some extra tools are necessary such as the shift-through operation and the fusion operation. Let us first explain the Givens-weight representation by an example.

The following graphical scheme represents the QR-factorization of a 6×6 Hessenberg matrix H, based on Givens transformations.



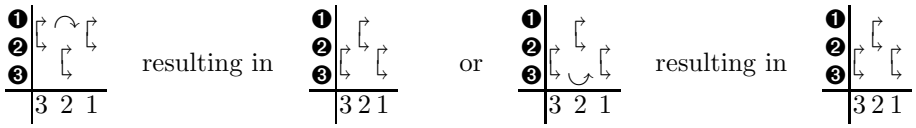
The scheme corresponds to the factorization $H = G_1 G_2 \dots G_5 R$. The matrix R is shown on the right by the $\times$ (assume them all to be different from zero) and is clearly upper triangular. The numbers on the vertical axis indicate the different rows of the matrix. The brackets with arrows depict Givens transformations. The arrows point to the rows the Givens transformations act on. The numbers on the bottom line represent a sort of timeline, describing the temporal order of application of each Givens transformation.

Givens transformations can also interact with each other by means of the *fusion* or the *shift-through* operations (see [18, pp. 112–115]). The fusion operation will be depicted as



and consists in the concatenation of two Givens transformations acting on the same rows. The shift-through operation allows us to rearrange the order of some Givens transformations (see [18]).

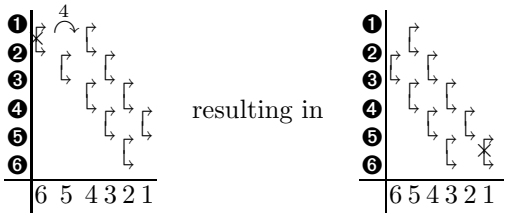
Graphically we will depict this rearrangement of Givens transformations as



The shift-through operation can also be applied repeatedly. This results in shift-through sequences. This is referred to as the shift-through operation of length ℓ.

LEMMA 2.1. *Suppose we have the matrix product GWX, in which G denotes a Givens transformation acting on rows* 1 *and* 2. *The matrices W and X are both unitary matrices consisting of a descending sequence of ℓ Givens transformations. This means that both W and X consist of ℓ successive Givens transformations. The ith Givens transformation G_i^W of W acts on rows $i+1$ and $i+2$. The ith Givens transformation G_i^X of X acts on rows i and $i+1$. The matrix product GWX can then be rewritten as $GWX = \hat{W}\hat{X}\hat{G}$, where $\hat{G}$ is now a Givens transformation acting on rows $\ell+1$ and $\ell+2$. The unitary matrices $\hat{W}$ and $\hat{X}$ are again descending sequences of ℓ Givens transformations.*

Graphically, the situation described in the previous lemma is



where G is the first Givens transformation acting on rows 1 and 2 in the first picture (marked with $\times$), W the leftmost chain of descending Givens, and X the rightmost chain of Givens. The symbol $\overset{\ell}{\curvearrowright}$ represents that after this operation, we will have a Givens transformation that will act on ℓ rows below the place where the initial Givens was acting. In this example, $\hat{G}$ will act on rows 5 and 6 because $\ell = 4$.

The operation depicted in the lemma is an operation "moving" a Givens rotation from the top-left position to the bottom-right position. Similarly one can have shift-through operations of length ℓ going to the top right, top left, and the bottom left.

To clearly indicate where the Givens is going and the number of single shift-through operations, we use the symbols $\overset{\ell}{\curvearrowright}$, $\overset{\ell}{\smile}$, $\overset{\ell}{\smile}$, and $\overset{\ell}{\curvearrowleft}$.

2.2. Similarity transform to Hessenberg form. Applying a QR-method directly to a Hermitian plus rank m matrix is possible, but, especially for large values of m, it is more efficient to first reduce the problem to Hessenberg form [6, 8, 3].

Suppose the initial matrix A is written as the sum of a Hermitian plus a rank m matrix, that is, $A = S + UV^H$, where U and $V \in \mathbb{C}^{n\times m}$. A standard reduction of the matrix A to Hessenberg form costs $\mathcal{O}(n^3)$ operations, when one does not exploit the available low rank structure. In particular, if S is a band matrix with bandwidth b, the cost reduces to $\mathcal{O}((b+m)n^2)$ [6, 3] for transforming the involved matrix A to Hessenberg form (see also [8, 9, 13]). The reduction to Hessenberg form preserves the eigenvalues since it is a unitary similarity transformation. This gives

$$H = Q^H AQ = Q^H SQ + Q^H UV^H Q = \hat{S} + \hat{U}\hat{V}^H.$$

Hence, the Hessenberg matrix H can be written as the sum of a Hermitian matrix $\hat{S}$ plus a rank m correction matrix $\hat{U}\hat{V}^H$. Note that, even if S was banded, $\hat{S}$ is a dense Hermitian matrix in general, but we will see that this matrix still admits an $\mathcal{O}(n)$ representation.

2.3. On the QR-algorithm. Let us denote the Hessenberg matrix we are working with as $H = S + UV^H$, where H is Hessenberg, S is Hermitian, and U and V are two $n \times m$ matrices. We will denote by $\mathcal{F}_{n,m}$ (see [7]) the class of upper Hessenberg matrices obtained as the sum of an $n \times n$ Hermitian matrix and a rank-m perturbation.

As shown in [7], the class $\mathcal{F}_{n,m}$ is closed under QR-steps. This fact is essential for developing an efficient implicit QR-method exploiting the matrix structure. The multishift QR-algorithm, applied to matrix $H = H^{(0)} \in \mathcal{F}_{n,m}$, generates a sequence of unitarily similar matrices $\{H^{(k)}\}_k$ converging to the real canonical Schur form of H.

The multishift QR-method proceeds iteratively as follows.[1] Let $H^{(0)} = H$,

$$\begin{cases} p_d^{(k)}(H^{(k)}) = Q^{(k)}R^{(k)}, \\ H^{(k+1)} = {Q^{(k)}}^H H^{(k)} Q^{(k)}, \quad k = 0, 1, \ldots, \end{cases} \tag{2.1}$$

where, for every k, $p_d^{(k)}(x)$ is a monic polynomial of degree d. For example, for $d = 1$ we have the single shift case with $p_1^{(k)}(H^{(k)}) = H^{(k)} - \mu^{(k)}I$, and for $d = 2$ we have the double-shift case with $p_2^{(k)}(H^{(k)}) = (H^{(k)} - \mu_1^{(k)}I)(H^{(k)} - \mu_2^{(k)}I)$. The parameter $\mu^{(k)}$ for the single shift case and parameters $\mu_1^{(k)}$ and $\mu_2^{(k)}$ for the double-shift case are chosen in accordance with some shift strategy [21, 23, 14] to accelerate convergence.

Usually, instead of explicitly computing the factorization QR of the polynomial in (2.1), we proceed implicitly, performing the transition $H^{(k)} \to H^{(k+1)}$ without even computing the products between the shifted matrices in (2.1) (see also [22]). The possibility of computing the iterations implicitly was first proposed in [11, 10]. Based on, e.g., the implicit Q-theorem [12], the essential uniqueness of the matrix obtained is guaranteed at each step, once the first column of $Q^{(k)}$ has been formed.

The global flow of an implicit method is hence of the following form.

[1] The superscript notation $\cdot^{(i)}$ will be used only for depicting QR-steps performed on matrices. We will omit this superscript as much as possible so as not to overload the notation.

1. *Initialization step.* Determine the orthogonal transformation $\hat{Q}_{\mathcal{I}}$ (the subscript $\mathcal{I}$ refers to the initial step), such that

$$\hat{Q}_{\mathcal{I}}^H p_d(H)\,\mathbf{e}_1 = \beta\,\mathbf{e}_1, \quad \beta = \|p_d(H)\mathbf{e}_1\|_2,$$

   where $p_d(H)$ is a monic polynomial of degree d, as described in (2.1) with suitable shifts chosen. Let $H_{\mathcal{I}} = \hat{Q}_{\mathcal{I}}^H H \hat{Q}_{\mathcal{I}}$, and note that $H_{\mathcal{I}}$ is not in upper Hessenberg form since a bulge is created with the tip in position $(d+2, 1)$.
2. *Chasing steps.* Perform a unitary similarity reduction on $H_{\mathcal{I}}$ to bring it back to Hessenberg form (not altering the first column and row). Let $\hat{Q}_{\mathcal{C}}$ (the subscript $\mathcal{C}$ refers to the chasing procedure) be such that $\hat{H} = \hat{Q}_{\mathcal{C}}^H H_{\mathcal{I}} \hat{Q}_{\mathcal{C}} = \hat{Q}_{\mathcal{C}}^H \hat{Q}_{\mathcal{I}}^H H \hat{Q}_{\mathcal{I}} \hat{Q}_{\mathcal{C}}$ is in Hessenberg form. Set $\hat{Q} = \hat{Q}_{\mathcal{I}} \hat{Q}_{\mathcal{C}}$.

It can be proved easily that $\hat{Q}$ and $\hat{H}$ are essentially identical to the matrices obtained by performing the explicit QR-algorithm. In this paper, we will consider only the double shift cases, that is, $d = 2$, but the proposed algorithm works similarly for the general case with any $d \geq 1$. The single shift case is fully treated in [20]. In the double shift case $p_2(H) = (H - \mu_1 I)(H - \mu_2 I)$ is a generalized Hessenberg and $Q_{\mathcal{I}}$ is obtained as the product of two Givens transformations. The size of the bulge created depends on the degree d and has $d(d+1)/2$ undesired elements. Finally, the orthogonal matrix $Q_{\mathcal{C}}$ in the chasing step is the cumulative product of $d\,(n-2)$ Givens factors.

3. A suitable $\mathcal{O}(n)$ representation. In [7] the matrices belonging to $\mathcal{F}_{n,m}$ are represented by using the diagonal and subdiagonal of H plus the two skinny matrices U and V. In this section we will deduce a representation of H based on Givens transformations similar to the one introduced for $m = 1$ in [15]. This representation still allows us to store $\mathcal{O}(n)$ elements and has the advantage of being intuitive, and stable, as usual, for representations based on Givens rotations (see, for example, [5, 17]).

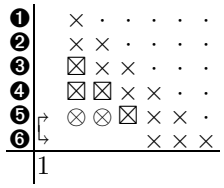
Since matrix H is the sum of a Hermitian and a low rank matrix, we know that the Hermitian matrix S has quite a large part of the low rank structure. Consider $S = H - UV^H$. The Hessenberg matrix H has zeros below the subdiagonal. Hence, matrix S needs to be of rank m below the subdiagonal. The matrix S has the following form, where the part marked with the $\boxtimes$ is coming from a rank-m part.

$$S = \begin{bmatrix} \times & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \cdot & \cdot & \cdot & \cdot \\ \boxtimes & \times & \times & \cdot & \cdot & \cdot \\ \boxtimes & \boxtimes & \times & \times & \cdot & \cdot \\ \boxtimes & \boxtimes & \boxtimes & \times & \times & \cdot \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \times & \times \end{bmatrix}$$

Since S is Hermitian, the entries denoted by dots are known once the lower triangular part of S is specified. The matrix we start with is the matrix S. All subsequent graphical schemes represent factorizations of this matrix S.

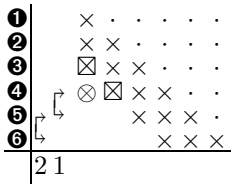
3.1. A rank-1 perturbation. We consider the case $m = 1$; hence, as we can see in the first scheme below, three elements (coming from a rank-1 part) in the last row of the matrix S are annihilated by a single Givens transformation acting on rows

5 and 6.

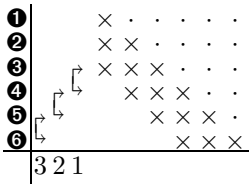


Mathematically, the scheme depicts $S = G_1 S_1$, where G_1 corresponds to the Givens transformation at time stamp 1, and S_1 is the matrix on the right having extra zeros in the last row. The scheme here presents a sort of factorization of the original Hermitian matrix S. In fact, S_1 and G_1 are computed similarly to how one computes a QR-factorization: $G_1^H S = S_1$, where G_1^H acts on S to create zeros, resulting in a matrix S_1.

The elements to be annihilated by the second Givens transformation are marked in the scheme with $\otimes$. In fact, G_2 is computed to obtain $G_2^H G_1^H S = S_2$, where S_2 has $n-4$ zeros in row $n-1$. Rewriting $G_2^H G_1^H S = S_2$ gives us $S = G_1 G_2 S_2$ depicted in the following scheme. We see S_2 on the right and Givens transformations G_1 above 2 and G_2 above 1. Considering $S = G_1 G_2 S_2$, we see that first G_2 needs to be reapplied to S_2 followed by G_1 to retrieve the original matrix S.



After the Givens transformation acting on rows three and four is performed, we have completely removed the low rank part. Removing the structured rank part introduces, however, a new subdiagonal. The matrix remaining on the right side is now a generalized Hessenberg matrix having two subdiagonals.



Let us write this factorization as $S = \tilde{V}\tilde{B}$, in which $\tilde{V}$ represents the Givens transformations on the left and the matrix $\tilde{B}$ represents the generalized Hessenberg matrix on the right. Reconsider now the original matrix H. Due to the equality between the low rank parts in the matrices $\mathbf{u}\mathbf{v}^H$ and S, we can factorize the vector $\mathbf{u}$ as $\tilde{V}\tilde{\mathbf{u}}$, where the vector $\tilde{\mathbf{u}}$ has only the first three elements different from zero. This means that we get

$$H = S + \mathbf{u}\mathbf{v}^H = \tilde{V}\left(\tilde{B} + \tilde{\mathbf{u}}\mathbf{v}^H\right) \tag{3.1}$$

$$= \quad \left(\begin{bmatrix} \times & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \cdot & \cdot & \cdot \\ & \times & \times & \times & \cdot & \cdot \\ & & \times & \times & \times & \cdot \\ & & & \times & \times & \times \end{bmatrix} + \begin{bmatrix} \times \\ \times \\ \times \\ \\ \\ \end{bmatrix} \mathbf{v}^H \right).$$

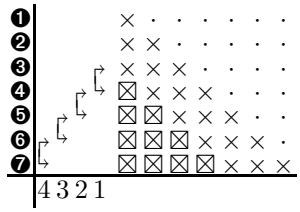
This representation was used in [15]. The disadvantage of this form is that it requires a quite complicated initial step. To simplify the QR-procedure we enlarge our matrix $\tilde{V}$ by adding two more Givens transformations to it. These transformations are constructed in such a way that they annihilate the second and third nonzero elements in $\tilde{\mathbf{u}}$; hence $\mathbf{u} = V\hat{\mathbf{u}}$. The following scheme depicts the representation we will use. Since there is no correspondence between the rank structure of $\tilde{B}$ and $\tilde{\mathbf{u}}$ anymore, no additional zeros will be created in the matrix $\tilde{B}$.

$$H = V\left(B + \hat{\mathbf{u}}\mathbf{v}^H\right)$$

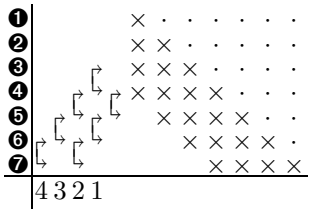
$$= \quad \left(\begin{bmatrix} \times & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \cdot & \cdot & \cdot \\ & \times & \times & \times & \cdot & \cdot \\ & & \times & \times & \times & \cdot \\ & & & \times & \times & \times \end{bmatrix} + \begin{bmatrix} \times \\ \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{v}^H \right).$$

For ease of notation we will reuse the symbol $\mathbf{u}$; that is, we start from a factorization $H = V(B + \mathbf{u}\mathbf{v}^H)$ assuming $\mathbf{u}$ has only one element different from zero.

3.2. A rank-2 perturbation. We assume the perturbation to be of rank 2; the general case proceeds similarly. Decompose UV^H as $\mathbf{u}\mathbf{v}^H + \mathbf{x}\mathbf{y}^H$. The Hermitian matrix S is of the form $S = H - \mathbf{u}\mathbf{v}^H - \mathbf{x}\mathbf{y}^H$, having therefore the part below the subdiagonal of rank at most 2. Similarly to what is done for the rank-1 case, we can peal off a rank-1 part. Now, this will not create zeros, because another part of rank 1 will still remain. Graphically a first sequence of Givens transformations transforms S into the following form, having the part marked with ⊠ still of rank 1. Without loss of generality we assume we are working on a 7×7 matrix.

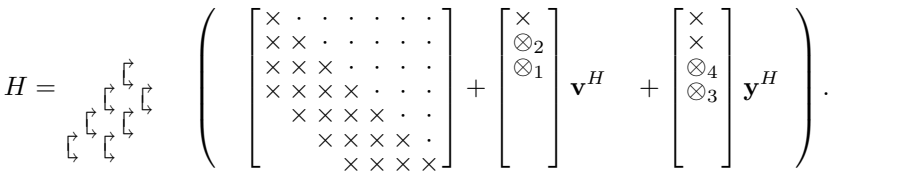


We remark that this first sequence of Givens transformations also transforms the vector $\mathbf{u}$ into a vector with only the first three elements nonzero. To remove the remaining low rank part in the scheme above, another sequence of Givens transformations is needed. We get the following form:



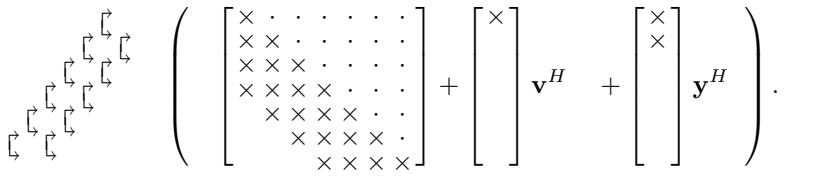
This second sequence of Givens transformations will also create many zeros in the vector $\mathbf{x}$; all but the first four elements will become zero. This means that for the

matrix H, we get the following graphical scheme:



Just like in the rank-one case, we are not finished yet. The elements $\otimes$ in the vectors still need to be removed. The order of their removal is indicated by their subscripts. Bringing these Givens transformations outside the brackets gives us the following final representation:

(3.2)



This scheme corresponds to (for simplicity we reuse the symbols $\mathbf{u}$ and $\mathbf{x}$ for denoting the sparse vectors) $H = V^{(\mathbf{u})}\, V^{(\mathbf{x})} \left(B + \mathbf{u}\mathbf{v}^H + \mathbf{x}\mathbf{y}^H\right)$.

Remark 1. We choose to present the construction of the sequences of Givens transformations $V^{(\mathbf{u})}$ and $V^{(\mathbf{x})}$ based on the matrix S, to show the appearance of the extra subdiagonals. In practice, however, they are entirely determined by the vectors $\mathbf{x}$ and $\mathbf{u}$, and one can immediately compute the full sequence of $n-1$ Givens transformations for $V^{(\mathbf{u})}$ and $n-2$ transformations for $V^{(\mathbf{x})}$ based on the two vectors $\mathbf{u}$ and $\mathbf{x}$.

3.3. The generic rank $m > 2$ perturbation. Generalizing, if $H = S + UV^H$, with UV^H a rank m perturbation of a Hermitian matrix S, we can represent H as

$$H = \prod_{i=1}^{m} V^{(\mathbf{u}_i)} \left(B + \sum_{i=1}^{m} \mathbf{u}_i \mathbf{v}_i^H \right),$$

where B is a generalized Hessenberg with $m+1$ subdiagonals, $\mathbf{u}_i$ are sparse vectors with only the first i entries different from zero, and each $V^{(\mathbf{u}_i)}$ consists of $n-i$ Givens transformations. So, to store the representation for a Hessenberg matrix which is the sum of a Hermitian plus rank m perturbation, we need to store

- $\sum_{i=1}^{m}(n-i) = \mathcal{O}(nm + m^2)$ Givens transformations;
- $\sum_{i=1}^{m+2}(n+1-i) = \mathcal{O}(nm + m^2)$ entries for the matrix B; and
- $\sum_{i=1}^{m} i = \mathcal{O}(m^2)$ entries for the vectors $\mathbf{u}_i$.

Based on this representation we will illustrate how to perform a single and double shift QR-step. We will discuss the single shift and multishift settings in two different sections, so the reader can follow the algorithms more easily. For simplicity we will restrict ourselves to the rank-two case and the double shift case. However, the ideas presented are easily generalizable to fit the more general case.

Remark 2. In section 4 we silently assume all Givens transformations in the sequences $V^{(\mathbf{u}_i)}$ to be different from the identity matrix. In this case all shift-through operations are well defined, and we can run the algorithm to completion. We will come back to this in the beginning of section 5 and discuss this in more detail in [20].

4. The multishift strategy. For simplicity we will present only the double shift case restricted to a rank-two perturbation; the general multishift case proceeds almost identically, and a detailed explanation of the single shift case is reported in [20]. The double shift case is quite important since one can restrict computations to the real field in case the original matrix A is non-Hermitian but real, by simply choosing the shifts as complex conjugates. In this case one converges not to an upper triangular matrix, but to a block upper triangular matrix having blocks of size at most 2×2 on the diagonal. The 2×2 blocks on the diagonal will then contain the complex conjugate eigenvalues of the original real-valued matrix A [21, 12].

The cost of a double shift QR-step is roughly double that of a single shift QR-step. Since, however, the convergence is assumed to be twice as fast, the global speed is approximately the same. Multishift algorithms have, however, many other advantages [21].

Let $H = H_1 = V_1^{(\mathbf{u})} V_1^{(\mathbf{x})} (B_1 + \mathbf{u}_1 \mathbf{v}_1^H + \mathbf{x}_1 \mathbf{y}_1^H)$. For performing a complete double shift QR-step on a Hessenberg matrix, $2(n-1)$ Givens transformations are needed. Two Givens transformations are needed in the initialization step, and two more are needed in each of the chasing steps to move the bulge down one row. In particular, at step i we have an "almost" Hessenberg matrix H_i. To go from H_i to H_{i+1} a similarity transformation is performed involving two Givens transformations $G_{i,2}$ and $G_{i,1}$. Transformation $G_{i,2}$ acts on rows i and $i+1$, and transformation $G_{i,1}$ acts on rows $i+1$ and $i+2$. First $G_{i,1}^H$ is applied, followed by $G_{i,2}^H$. This means that $H_{i+1} = G_{i,2}^H G_{i,1}^H H_i G_{i,1} G_{i,2}$. The vectors and matrices used for representing H_i will inherit the same subscript as well as all intermediate variables with the subscript i, unless they are in final form for H_{i+1}; in that case they obtain the subscript $i+1$. In particular, at the end of the initialization step we will end up with matrix H_2, which is a perturbed Hessenberg with a bulge of size 3. This is matrix $H_{\mathcal{I}}$ from section 2.3.

4.1. Initialization. Let $\hat{Q}_{\mathcal{I}}$ be the initial unitary transformation such that

$$\hat{Q}_{\mathcal{I}}^H (H - \mu_1 I)(H - \mu_2 I)\mathbf{e}_1 = \beta\, \mathbf{e}_1, \qquad \text{where} \qquad \beta = \|(H - \mu_1 I)(H - \mu_2 I)\mathbf{e}_1\|_2.$$

The orthogonal transformation $\hat{Q}_{\mathcal{I}}^H$ consists of two Givens transformations. In particular, $Q_{\mathcal{I}} = G_{1,1} G_{1,2}$ such that $G_{1,2}^H G_{1,1}^H (H - \mu_1 I)(H - \mu_2 I)\mathbf{e}_1 = \beta\, \mathbf{e}_1$. We start by applying a similarity transformation on $H = H_1$, i.e., the initial step. We obtain

$$H_2 = G_{1,2}^H G_{1,1}^H\, H_1\, G_{1,1} G_{1,2} = G_{1,2}^H G_{1,1}^H\, V_1^{(\mathbf{u})} V_1^{(\mathbf{x})} (B_1 + \mathbf{u}_1 \mathbf{v}_1^H + \mathbf{x}_1 \mathbf{y}_1^H)\, G_{1,1} G_{1,2}$$

$$= \quad \left(\begin{bmatrix} \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \times & \cdot & \cdot & \cdot \\ & \times & \times & \times & \times & \cdot & \cdot \\ & & \times & \times & \times & \times & \cdot \\ & & & \times & \times & \times & \times \end{bmatrix} + \begin{bmatrix} \times \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{v}_1^H + \begin{bmatrix} \times \\ \times \\ \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{y}_1^H \right) G_{1,1} G_{1,2}. \tag{4.1}$$

The Givens transformations depicted on the left, marked with $\times$, are the transformations $G_{1,2}^H$ and $G_{1,1}^H$.

4.1.1. Transformations on the left. Similarly as in the single shift case presented in [20], we shift the transformations on the left through the sequences $V_1^{(\mathbf{u})}$ and $V_1^{(\mathbf{x})}$ as depicted in scheme (4.1). Based on the equality $G_{1,2}^H G_{1,1}^H\, V_1^{(\mathbf{u})} V_1^{(\mathbf{x})} = \tilde{V}_1^{(\mathbf{u})} \tilde{V}_1^{(\mathbf{x})}\, \tilde{G}_{1,2}^H \tilde{G}_{1,1}^H$, we obtain

$$H_2 = \tilde{V}_1^{(\mathbf{u})}\tilde{V}_1^{(\mathbf{x})}\,\tilde{G}_{1,2}^H\tilde{G}_{1,1}^H(B_1 + \mathbf{u}_1\mathbf{v}_1^H + \mathbf{x}_1\mathbf{y}_1^H)\,G_{1,1}G_{1,2}$$

$$= \quad \left(\begin{bmatrix} \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \times & \cdot & \cdot & \cdot \\ & \times & \times & \times & \times & \cdot & \cdot \\ & & \times & \times & \times & \times & \cdot \\ & & & \times & \times & \times & \times \end{bmatrix} + \begin{bmatrix} \times \\ \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{v}_1^H \quad + \begin{bmatrix} \times \\ \times \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{y}_1^H \right) G_{1,1}G_{1,2}.$$

Performing the operations $\tilde{G}_{1,2}^H\tilde{G}_{1,1}^H$ on the terms in the brackets affects only the matrix B_1. Due to the zero structure of the vectors $\mathbf{u}_1$ and $\mathbf{x}_1$, they remain unaltered and are already in final form ($\mathbf{u}_2 = \mathbf{u}_1$ and $\mathbf{x}_2 = \mathbf{x}_1$):

$$H_2 = \tilde{V}_1^{(\mathbf{u})}\tilde{V}_1^{(\mathbf{x})}\,(\tilde{B}_1 + \mathbf{u}_2\mathbf{v}_1^H + \mathbf{x}_2\mathbf{y}_1^H)\,G_{1,1}G_{1,2}$$

$$= \quad \left(\begin{bmatrix} \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \times & \cdot & \cdot & \cdot \\ \hat{\otimes} & \times & \times & \times & \times & \cdot & \cdot \\ & \times & \times & \times & \times & \cdot & \\ & & \times & \times & \times & \times & \end{bmatrix} + \begin{bmatrix} \times \\ \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{v}_1^H \quad + \begin{bmatrix} \times \\ \times \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{y}_1^H \right) G_{1,1}G_{1,2}.$$

One undesired element is created in the matrix $\tilde{B}_1$. This element is marked with $\hat{\otimes}$ since it will play a role in the chasing. In addition to the Givens transformations moving to the bottom, this bulge will move downward as the iteration proceeds.

4.1.2. Transformations on the right. As we will see in the following, the remaining transformations on the right are almost identical to the chasing procedure. Performing the transformations on the right on the terms inside the brackets gives us

$$H_2 = \tilde{V}_1^{(\mathbf{u})}\tilde{V}_1^{(\mathbf{x})}\,(\hat{B}_1 + \mathbf{u}_2\mathbf{v}_2^H + \mathbf{x}_2\mathbf{y}_2^H)$$

$$= \quad \left(\begin{bmatrix} \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \times & \cdot & \cdot & \cdot \\ \otimes & \times & \times & \times & \times & \cdot & \cdot \\ \otimes & \otimes & \times & \times & \times & \times & \cdot \\ & & & \times & \times & \times & \times \end{bmatrix} + \begin{bmatrix} \times \\ \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{v}_2^H \quad + \begin{bmatrix} \times \\ \times \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{y}_2^H \right).$$

As mentioned before, the bulge plays a role in the chasing and will move down along this subdiagonal. Hence, we attempt only to remove the undesired elements in the first column of the matrix $\hat{B}_1$. Two Givens transformations are needed.

$$H_2 = \tilde{V}_1^{(\mathbf{u})}\tilde{V}_1^{(\mathbf{x})}\hat{G}_{1,1}\hat{G}_{1,2}(B_2 + \mathbf{u}_2\mathbf{v}_2^H + \mathbf{x}_2\mathbf{y}_2^H)$$

$$= \quad \left(\begin{bmatrix} \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \times & \cdot & \cdot & \cdot \\ & \times & \times & \times & \times & \cdot & \cdot \\ & \hat{\otimes} & \times & \times & \times & \times & \cdot \\ & & & \times & \times & \times & \times \end{bmatrix} + \begin{bmatrix} \times \\ \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{v}_2^H \quad + \begin{bmatrix} \times \\ \times \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{y}_2^H \right).$$

Performing shift-through operations to bring the transformations $\hat{G}_{1,1}\hat{G}_{1,2}$ to the front gives us $\tilde{V}_1^{(\mathbf{u})}\tilde{V}_1^{(\mathbf{x})}\hat{G}_{1,1}\hat{G}_{1,2} = G_{2,1}G_{2,2}V_2^{(\mathbf{u})}V_2^{(\mathbf{x})}$, resulting in

$$H_2 = G_{2,1}G_{2,2}V_2^{(\mathbf{u})}V_2^{(\mathbf{x})}(B_2 + \mathbf{u}_2\mathbf{v}_2^H + \mathbf{x}_2\mathbf{y}_2^H), \tag{4.2}$$

which finalizes the initialization step.

4.2. The chasing. The global flow of the chasing is as follows. When starting the chasing procedure we have a matrix factorization of the form (assume $i < n-3$)

$$H_i = G_{i,1}G_{i,2}V_i^{(\mathbf{u})}V_i^{(\mathbf{x})}(B_i + \mathbf{u}_i\mathbf{v}_i^H + \mathbf{x}_i\mathbf{y}_i^H),$$

where all vectors and matrices satisfy the constraints posed by the representation, only the matrix B_i has a bulge because of a nonzero element in position $(i+2, i)$. One starts with a similarity transformation determined by $G_{i,1}G_{i,2}$.

$$\begin{aligned} H_{i+1} &= G_{i,2}^H G_{i,1}^H H_i G_{i,1} G_{i,2} \\ &= V_i^{(\mathbf{u})}V_i^{(\mathbf{x})}(B_i + \mathbf{u}_{i+1}\mathbf{v}_i^H + \mathbf{x}_{i+1}\mathbf{y}_i^H)G_{i,1}G_{i,2} \\ &= V_i^{(\mathbf{u})}V_i^{(\mathbf{x})}(\hat{B}_i + \mathbf{u}_{i+1}\mathbf{v}_{i+1}^H + \mathbf{x}_{i+1}\mathbf{y}_{i+1}^H). \end{aligned}$$

The vectors $\mathbf{u}_i = \mathbf{u}_{i+1}$ and $\mathbf{x}_i = \mathbf{x}_{i+1}$ are not affected by the transformations. The matrix $\hat{B}_i$ has three undesired nonzero elements in positions $(i+2, i)$, $(i+3, i)$, and $(i+3, i+1)$. The Givens transformations $\hat{G}_{i+1,1}^H$ and $\hat{G}_{i+1,2}^H$ are constructed so as to annihilate, respectively, the elements in positions $(i+3, i)$ and $(i+2, i)$:

$$H_{i+1} = V_i^{(\mathbf{u})}V_i^{(\mathbf{x})}\hat{G}_{i,1}\hat{G}_{i,2}(B_{i+1} + \mathbf{u}_{i+1}\mathbf{v}_{i+1}^H + \mathbf{x}_{i+1}\mathbf{y}_{i+1}^H).$$

The matrix B_{i+1} has only one undesired nonzero element in position $(i+3, i+1)$, which is hence shifted down with respect to the position in B_i. Applying the shift-through operation a few times to bring the Givens transformations to the front gives us

$$H_{i+1} = G_{i+1,1}G_{i+1,2}V_{i+1}^{(\mathbf{u})}V_{i+1}^{(\mathbf{x})}(B_{i+1} + \mathbf{u}_{i+1}\mathbf{v}_{i+1}^H + \mathbf{x}_{i+1}\mathbf{y}_{i+1}^H).$$

4.3. The final similarity transformations in the chasing. The implicit algorithm cannot be carried on fully to the end. Suppose we have the matrix H_{n-5}, and we perform the similarity transformation determined by Givens transformations $G_{n-5,1}$ and $G_{n-5,2}$: $H_{n-4} = G_{n-5,2}^H G_{n-5,1}^H H_{n-5} G_{n-5,2} G_{n-5,1}$. Applying the Givens transformations on the right, which act on columns from $n-5$ to $n-2$, will not create a bulge in the matrix B_{n-5}. Hence, we cannot continue the chasing procedure. The algorithm breaks down in this case. To determine the final Givens transformations, we need to explicitly compute a few columns of H_{n-5}. Using the representation this can be achieved easily (see [20] for the details).

5. Deflation. The QR-iterations should always be combined with a deflation technique, which allows us to split the problem into two or more smaller problems with the same structure. For a Hessenberg matrix H, deflation means that the Hessenberg matrix is numerically of the form

$$H = \left[\begin{array}{c|c} H(1:i,1:i) & \times \\ \hline 0 & H(i+1:n,i+1:n) \end{array}\right] = \left[\begin{array}{ccccccc|ccccccc} \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \hline & & & & & & 0 & \times & \times & \times & \times & \times & \times & \times \\ & & & & & & & \times & \times & \times & \times & \times & \times & \times \\ & & & & & & & & \times & \times & \times & \times & \times & \times \\ & & & & & & & & & \times & \times & \times & \times & \times \\ & & & & & & & & & & \times & \times & \times & \times \\ & & & & & & & & & & & \times & \times & \times \end{array}\right].$$

The matrix can be subdivided into two other Hessenberg matrices $H(1:i, 1:i)$ and $H(i+1:n, i+1:n)$ on which one can continue working separately.

In this paper, however, there is a strong connection between the Hermitian matrix S and the rank-m part. The matrix H has a zero block, but this does not necessarily mean that either S or UV^H has a zero block in the corresponding position. Only their sum must have a zero block in this position.

The coupled representation used in this paper makes it impossible to apply deflation in the classical sense. This means that one cannot just divide the Hessenberg matrix into two parts, each having their specific representation. In this case we maintain a global representation for the entire matrix, but the knowledge that there are deflatable blocks will make it possible to act on these different blocks separately.

In [20] we treat the algorithmic changes for accounting of special structures in the low rank perturbation.

5.1. The representation in case of deflation.

5.1.1. Assumptions on the Givens transformations. In the remainder of this section we assume a strict rank-two perturbation. This means that the vectors $\mathbf{u}$ and $\mathbf{x}$ have no trailing zeros. No trailing zeros means that the sequences of Givens transformations $V^{(\mathbf{u})}$ and $V^{(\mathbf{x})}$ consist of nonidentity Givens transformations.[2]

Since the shift-through operation of three nonidentity Givens transformations always results in three new nonidentity Givens transformations, the chasing procedure can never create identity Givens transformations in the middle of the sequences $V^{(\mathbf{u})}$ or $V^{(\mathbf{x})}$. The creation of identity Givens transformations can occur only at the left part of the sequence, when, for example, a fusion is applied. We can therefore conclude that after a QR-step is performed, we can have Givens transformations equal to the identity matrix only at the bottom (leftmost) part of the sequence. In this case again we refer the reader to [20]. So, to conclude, we explicitly assume all Givens transformations in the sequences to be different from the identity.

5.1.2. Detection of deflation. The easiest, but not the fastest, way to detect numerical zeros on the subdiagonal of the Hessenberg matrix is to explicitly compute these subdiagonal elements. This approach is used in [7]. Due to our specific representation, there is, however, a faster way to detect deflation that in the case $m = 0$ coincides with the standard approach.

Suppose the matrix $H = V^{(\mathbf{u})}V^{(\mathbf{x})}\left(B + \mathbf{u}\mathbf{v}^H + \mathbf{x}\mathbf{y}^H\right)$ we are working with can be split into two independent problems. This means that the element $H(i+1) = 0$ for a specific i.

Let us denote $W = {V^{(\mathbf{x})}}^H {V^{(\mathbf{u})}}^H H = B + \mathbf{u}\mathbf{v}^H + \mathbf{x}\mathbf{y}^H$. Since the matrices $V^{(\mathbf{u})}$ and $V^{(\mathbf{x})}$ represent sequences of ascending Givens transformations, we have $H(i+1, i) = 0$ if and only if $W(i+3, i) = 0$. Since $i + 3 \geq 4$, we have $\mathbf{u}(4 : n) = 0$ and $\mathbf{x}(4 : n) = 0$. This proves that $W(i+3, i) = 0$ if and only if $B(i+3, i) = 0$ because all the Givens transformations are different from the identity. The condition $B(i+3, i) = 0$ can be checked by using a relative criterion involving the size of the neighboring elements, making it easy to verify whether deflation can be applied.

The representation, however, does not reveal deflation in two other cases. In fact, we cannot determine if $H(n-1, n-2) = 0$ or $H(n, n-1) = 0$ without explicitly computing these entries, retrieving the values from the representation. This problem is similar to the problem we had for computing the final steps in the chasing steps of the QR-method (see section 4.3).

[2] Assuming, e.g., $\mathbf{u}$ to have a trailing sequence of zeros results in a leading (leftmost) sequence of identity Givens transformations in $V^{(\mathbf{u})}$. This is excluded in this paper but is discussed in [20].

Graphically, $H(i+1,i)=0$ corresponds to a factorization of the form (the dashed line depicts where deflation should be applied in the original Hessenberg matrix H)

(5.2)

$$H = \begin{pmatrix} \begin{bmatrix} \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \times & \cdot & \cdot & \cdot \\ \hdashline \times & \times & \times & 0 & \times & \cdot & \cdot \\ & \times & \times & & \times & \times & \cdot \\ & & & & \times & \times & \times \end{bmatrix} + \begin{bmatrix} \times \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{v}_2^H + \begin{bmatrix} \times \\ \times \\ \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{y}_2^H \end{pmatrix}.$$

Note that the zero indicating deflation is positioned two rows below the position where one should actually subdivide the matrix; the column, however, remains the same.

Remark 3. Considering a rank-m perturbation, we have $m+1$ subdiagonals. We assume all used Givens transformations to be different from the identity (otherwise we refer the reader to [20]). A zero element in the Hessenberg matrix can be detected as a zero on the $(m+1)$th subdiagonal. The zero resides in the same column but shifts down m rows. The final m subdiagonal elements of the Hessenberg matrix have to be computed explicitly; their zeroness is not reflected in the representation simply because we cannot shift down m more rows.

In the standard deflation approach, the matrix is divided into two independent subproblems with the same characteristics of the original one. On the contrary, in our case, the problem cannot be decoupled into two smaller subproblems because we are using a different representation and the two parts are interacting.

5.2. Performing *QR*-steps after deflation was applied. In the previous section it was shown that even though deflation can be applied, both independent Hessenberg matrices (see (5.1)) $H(1:i,1:i)$ and $H(i+1:n,i+1:n)$ still have interaction in the representation.

In the following subsections we will show how to perform QR-steps on the top parts $H(1:i,1:i)$, the bottom parts $H(i:n,i:n)$ $(1<i<n)$, and also the middle parts $H(i:j,i:j)$, $1<i<j<n$ (assuming, of course, suitable deflation possibilities in H for each of the three cases). We denote the entire matrix we act on as H; this corresponds to the matrix H from (5.1) and (5.2). We will, however, work only on the top, bottom, and middle parts of H!

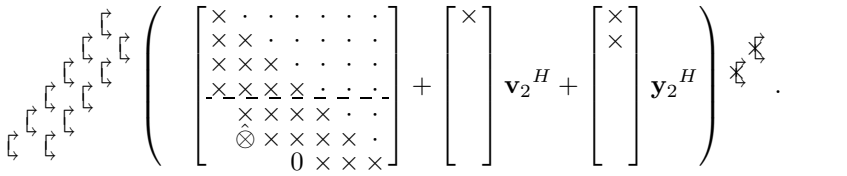
5.2.1. The top part. Assume deflation is possible and the following matrix factorization is given (the dashed line depicts the actual splitting between the upper and lower parts in the Hessenberg matrix). We start performing a QR-step on the matrix $H(1:i,1:i)$. Denote H by H_1, that is, $H_1=H$.

$$H_1 = \begin{pmatrix} \begin{bmatrix} \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \times & \times & \cdot & \cdot & \cdot \\ \hdashline & \times & \times & \times & \times & \cdot & \cdot \\ & & \times & \times & \times & \times & \cdot \\ & & & 0 & \times & \times & \times \end{bmatrix} + \begin{bmatrix} \times \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{v}_1{}^H + \begin{bmatrix} \times \\ \times \\ \\ \\ \\ \\ \\ \end{bmatrix} \mathbf{y}_1{}^H \end{pmatrix}.$$

When performing a QR-step on the top part of the matrix, one would expect the chasing step to break down once the dashed line is reached. Assume the shifts are chosen with regard to rapid convergence for the upper part of the Hessenberg matrix.[3]

The initial transformations are performed similarly as in the standard case. Hence we obtain (4.2): $H_2 = G_{2,1}G_{2,2}V_2^{(\mathbf{u})}V_2^{(\mathbf{x})}(B_2 + \mathbf{u}_2\mathbf{v}_2^H + \mathbf{x}_2\mathbf{y}_2^H)$, with $G_{2,1}$ and $G_{2,2}$ determining the upcoming chasing step. Performing the unitary similarity transformation with $G_{2,2}$ and $G_{2,1}$ gives us $H_3 = V_2^{(\mathbf{u})}V_2^{(\mathbf{x})}(B_2 + \mathbf{u}_2\mathbf{v}_2^H + \mathbf{x}_2\mathbf{y}_2^H)G_{2,1}G_{2,2}$, graphically depicted as follows ($G_{2,1}$ and $G_{2,2}$ are marked with $\times$ and shown on the right):

$H_2 =$

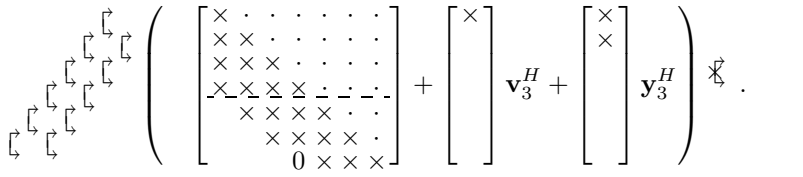


We see that since $B(7,4) = 0$, no bulge is created when transformation $G_{2,1}$ is applied to the columns; multiplication by matrix $G_{2,2}$ will leave a small bulge in position $(6,2)$ that can be removed by means of a Givens transformation acting on rows 5 and 6. This transformation can simply be dragged to the left as was done before by repeatedly applying the shift-through operation. It is interesting to remark that we have an impact below the dashed line! We obtain

$$H_3 = G_{3,1}V_3^{(\mathbf{x})}V_3^{(\mathbf{u})}\left(B_3 + \mathbf{u}_3\mathbf{v}_3^H + \mathbf{x}_3\mathbf{y}_3^H\right).$$

Another similarity transformation is now performed, involving Givens transformation $G_{3,1}$. Since $H(5,4) = 0$, we would expect the algorithm to break down after this similarity transformation. Before applying $G_{3,1}$ on the right we have

(5.3) $H_3 =$

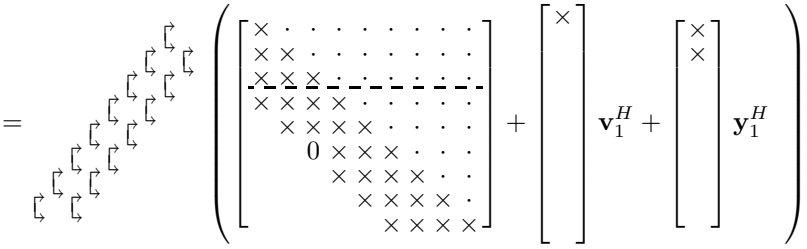


After applying $G_{3,1}$ on the right nothing essential changes. When trying to perform the chasing step, we see that no bulge in position $(7,3)$ will be created and the algorithm breaks down, as expected. One has performed an implicit QR-step on $H(1:i,1:i)$.

5.2.2. Bottom part. Assume we have a matrix $H = H_1$ and we would like to perform a QR-step now on the bottom part $H(i:n,i:n)$. Even though the Givens transformations acting on the top part will be of no influence, just as the vectors $\mathbf{x}$ and $\mathbf{u}$, we do depict them to avoid confusion. For simplicity we restrict our attention

[3]The shifts are therefore computed based on the bottom right part of the matrix $H(1:i,1:i)$.

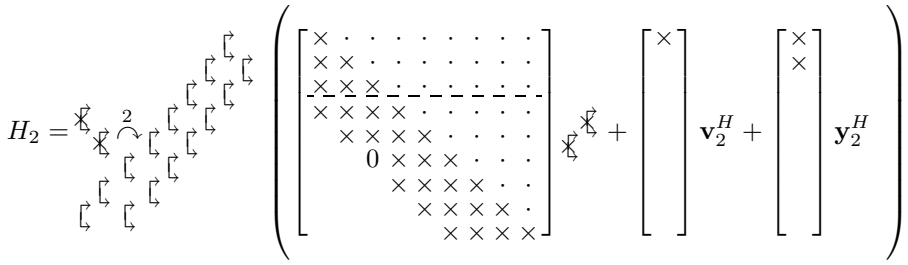
to a 9×9 matrix; the general case proceeds similarly.

(5.4) $H_2 =$ 

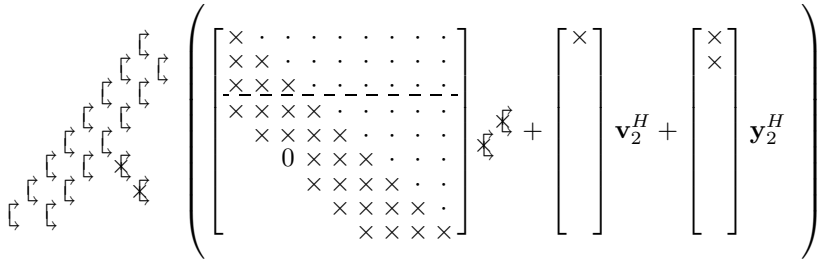
The shifts are computed and the initial Givens transformations are determined by explicitly computing the necessary columns of the matrix H. With respect to the original matrix in (5.1), we are now choosing two shifts μ_1 and μ_2 and two Givens transformations such that

$$G_{1,2}^H G_{1,1}^H (H(i:n,i:n) - \mu_1 I)(H(i:n,i:n) - \mu_2 I)\mathbf{e}_1 = \beta \mathbf{e}_1,$$

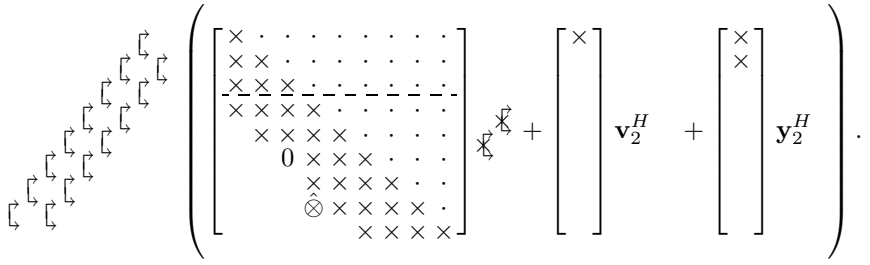
where $\mathbf{e}_1$ has the size of $H(i:n,i:n)$. The two Givens transformations should act on rows $1,2$ and $2,3$ of the matrix $H(i:n,i:n)$, but since we are working with H, which has extra rows on top, $G_{1,2}^H$ is acting on row 4 and 5 while $G_{1,1}^H$ combines row 5 with row 6. Graphically, we obtain (the Givens rotations determining the similarity transformation are marked with $\times$)

(5.5) $H_2 =$ 

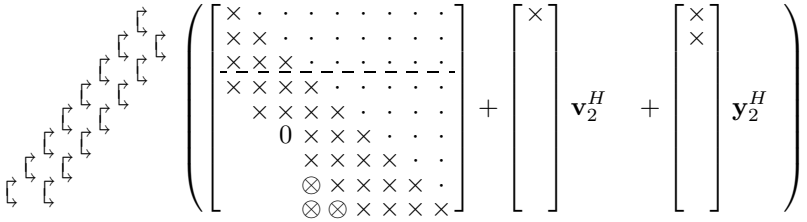
We expect that the depicted zero remains in this position after the initial similarity transformation is performed; otherwise our initial deflation was destroyed. In fact, we can do almost exactly the same thing as in the standard initial step. We first shift through the leftmost marked Givens transformations, as depicted in (5.5). We obtain the following situation:

$H_2 =$ 

Now the zero plays an important role. Applying these two left transformations on the matrix B_1 will create only one bulge. Hence the zero remains intact.

$H_2 =$ 

There are two Givens transformations remaining on the right. Applying them, we will produce a bulge:

$H_2 =$ 

To remove the bulge we have to combine rows 8 and 9 and rows 7 and 8. These last two Givens transformations can be brought to the front by applying a series of shift-through operations (just like in section 4.1), and one can continue the chasing procedure.

5.2.3. Middle part. When running the algorithm zeros will be created in many more instances. This will also result in cases, such as, for example,

$$H = \left[\begin{array}{c|c|c} H(1:i-1,1:i-1) & \times & \times \\ \hline 0 & H(i:j,i:j) & \times \\ \hline 0 & 0 & H(j+1:n,j+1:n) \end{array}\right]. \tag{5.6}$$

We already know how to perform QR-steps on the top part, the matrix $H(1:i-1,1:i-1)$ (standard initial step and new finishing step), and on the bottom part, matrix $H(j+1:n,j+1:n)$ (standard initial step and standard finishing step). In fact, performing a QR-step on the middle part is just a combination of the standard initial step with the new finishing step from the top part.

6. Software and experiments.

6.1. The software. In this section we present some details concerning the software implementation. The code was written both in MATLAB R2007b and in Fortran 95 on an Intel Core 2 Duo 2.4 GHz with 4GB of memory, and both implementations are available online.[4]

As described in section 3, the representation is based on 2×2 Givens rotations, which can be stored by either one or two parameters. So globally the memory used by the algorithm for the rank-one case is given by $n-1$ places for the unitary 2×2 matrices,[5] $3(n-1)$ places for the diagonal, subdiagonal, and subsubdiagonal of the

[4] http://www.cs.kuleuven.be/∼raf/

[5] In the actual implementation we used 2×2 unitary matrices instead of Givens transformations; these matrices, although more expensive in the occupation of memory, are easier to handle.

matrix B, and finally n places for the product $\mathbf{u}\mathbf{v}^H$. For the rank-two perturbation, an extra sequence of unitary transformations, an extra subdiagonal for B, and an extra vector for the product $\mathbf{x}\mathbf{y}^H$ are needed.

Since, in general, we will consider non-Hermitian eigenvalue problems, the Rayleigh shift was taken. This means that, for the single shift case, we chose the shift value equal to the last diagonal element, while for the double shift case, both the eigenvalues of the trailing 2×2 block were taken.

It is well known that balancing can have a significant impact on the accuracy of the computed results. This means that a permuted diagonal matrix D is constructed such that the matrix $D^{-1}AD$ has the row and column norms as nearly as possible equal to each other. See, e.g., the matrix in (6.2), which becomes Hermitian tridiagonal and hence has perfectly conditioned eigenvalues after scaling with the matrix $D = \text{diag}([1, \ldots, 1, \sqrt{\alpha}])$. Although, in the general, non-Hermitian setting, balancing can be performed without bothering about the structure, in our case, the Hermitian plus low rank structure needs to be maintained; hence it is not trivial to design a good balancing strategy maintaining this structure.

6.1.1. Deflation. For a generic Hessenberg matrix $H = (h_{ij})$, deflation is admitted when the following constraint is satisfied (see, e.g., [23]):

$$h_{i+1,i} < \epsilon(|h_{ii}| + |h_{i+1,i+1}|), \tag{6.1}$$

ϵ being the machine precision.

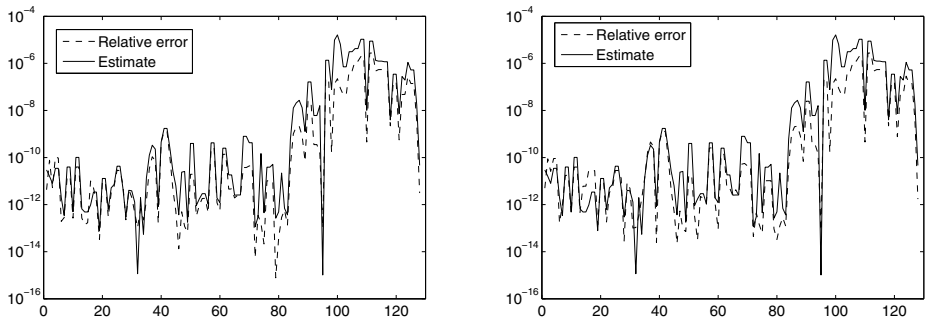
This criterion is, however, based on the entire matrix H, which consists of a sum of a Hermitian and a low rank matrix. Both the Hermitian and the low rank matrix have, however, a nonneglectable low rank part below the subdiagonal. Perfect annihilation is required in order to create zeros in the matrix H. Numerical experiments show that in case of coefficients of largely varying sizes in the matrix H, this annihilation might not be perfect anymore. Round-off introduces errors, making it difficult/impossible to accurately compute the subdiagonal elements $h_{i+1,i}$. The other matrix elements in H are computed up to high precision, but the difficulties of determining $h_{i+1,i}$ sometimes result in too early or too late termination of the QR-steps, resulting often in a loss of accuracy. This behavior was encountered in several of the numerical experiments performed.

In section 5.1.2, it was illustrated how a zero in H can be detected in the representation by finding a corresponding zero in a subdiagonal of the matrix B. Let us refer to these two deflation criteria as the $\mathcal{H}$-criterion (Hessenberg) or the $\mathcal{B}$-criterion (zero detected in the matrix B). For detecting a zero in a subdiagonal of B we use (6.1), but then applied to the subsubdiagonal elements of B.

In the following numerical experiment we compare both deflation criteria for the same matrix of size 128. In both figures all eigenvalues are plotted, and their estimate of accuracy based on the condition number is also depicted. Figure 6.1(a) represents the eigenvalues computed via the standard $\mathcal{B}$-deflation criterion, while in Figure 6.1(b) we used the $\mathcal{H}$-criterion. Both figures look almost identical, and relative accuracy of all of the eigenvalues is comparable.[6] The average number of iterations is significantly different. The $\mathcal{B}$-criterion uses 4.5 iterations on average, whereas the $\mathcal{H}$-criterion due to cancellation needs on average 14 iterations. In fact, the iteration process can be stopped far before the 14th iteration, but one is not able to determine the subdiagonal

[6] In this example all elements were constructed randomly. When sizes of the elements vary more, accuracy is quite often lost in the $\mathcal{H}$-criterion.

elements accurately, and hence one is not sure when to terminate the iteration process to obtain a certain level of accuracy.



(a) $\mathcal{B}$-criterion (b) $\mathcal{H}$-criterion

FIG. 6.1. *Comparison of two deflation criteria. Relative error in the eigenvalues and estimated error using the $\mathcal{B}$-criterion* (a) *and the $\mathcal{H}$-criterion* (b).

6.1.2. Computational complexity. An important operation in the algorithm is the shift-through operation. Each shift-through operation involves a QR-factorization of a 3×3 unitary matrix. This involves computing the two Givens transformations and takes in total 32 operations.

Performing a simple QR-step can be divided in two main steps:

- First operation on the left. This involves a single shift-through operation (32 flops), followed by applying the unitary transformation on the matrix B (≈ 18 flops).
- Inside the chasing, one has to perform the unitary transformation on the right and drag it through the matrix B such that it pops up again on the left. This has to be performed $n-1$ times. To perform the operations, one needs a window of 4×3 sliding on the main diagonal of the matrix B. Computing this window takes ≈ 30 operations. Applying the unitary matrix on $\mathbf{v}$ takes 6 operations; applying it on B and factoring it again out takes ≈ 48 flops. Performing the shift-through operation to bring it to the front takes ≈ 32 operations.[7]

Hence we obtain a global cost of approximately $116n-66$ for each QR-step on a Hessenberg matrix of size n. The designed algorithm is hence clearly an $\mathcal{O}(n^2)$ method.

6.2. The numerical experiments. In this section we present the results of the numerical experiments performed on matrices belonging to the class $\mathcal{F}_{n,1}$.

We denote by $\boldsymbol{\lambda}=[\lambda_1,\lambda_2,\ldots,\lambda_n]$ the vector containing the exact eigenvalues and by $\tilde{\boldsymbol{\lambda}}=[\tilde{\lambda}_1,\tilde{\lambda}_2,\ldots,\tilde{\lambda}_n]$ the vector of the computed ones, sorted in such a way that $\tilde{\lambda}_i$

[7]In this rough estimate we neglect the final step, which needs to be determined explicitly; since no shift-through needs to be performed, the cost is roughly the same.

is an approximation of λ_i. The error criteria for measuring accuracy are as follows:

$$E^{(abs)} = \|\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}\|_\infty = \max_i \left\{ |\lambda_i - \tilde{\lambda}_i| \right\},$$

$$E^{(rel)} = \max_i \left\{ \frac{|\lambda_i - \tilde{\lambda}_i|}{|\lambda_i|} \right\}.$$

To compare with the results from [7] we also use the infinity norm relative criterion defined as

$$E^{(rel2)} = \frac{\|\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}\|_\infty}{\|\boldsymbol{\lambda}\|_\infty}.$$

Note that $E^{(rel)} <$ tol guarantees that all the eigenvalues have been computed with a relative error lower than the tolerance tol, while this is not true when $E^{(rel2)}$ is considered. In our numerical results we compared the eigenvalues returned by our program (denoted as `GivQR`) with the eigenvalues returned by the Lapack routine `DGEEV`, and we used a cutting criterion of 10^{-16} as deflation tolerance.

The purpose of this experimentation is to test accuracy in the eigenvalues' computation with respect to general purpose algorithms not specifically designed to deal with structured matrices. The test suite consists of different kinds of perturbed tridiagonal matrices and of Hessenberg forms of Hermitian plus low rank matrices. The results of a more extensive testing are reported in [20].

6.2.1. Description of the test suite. In this section we describe the matrices used to estimate the quality of our algorithm, both from the point of view of accuracy and computational speed. We were able to compare with a general purpose method implemented in Lapack, showing that already for matrices of moderate size the time required by our implementation is lower than that taken by the `DGEEV` routine in Lapack. The behavior of our method in terms of CPU time differs from the `FastQR` algorithm by Eidelman, Gemignani, and Gohberg [7] for quantities of the order of milliseconds, so we do not report the direct comparison in terms of time because they behave similarly.

Many of the matrices considered have a particular interest from an applicative point of view. In particular, we tested our software on almost tridiagonal matrices and on Hessenberg matrices that can be expressed as the sum of Hermitian and rank-one matrices. Changing slightly the notation introduced in [7], we will consider these different test matrices.

Type I: *Chebyshev–Comrade matrices* are matrices of the form $A = T_n + \mathbf{u}\mathbf{e_n}^T$, where T_n is an $n \times n$ Hermitian tridiagonal matrix associated with Chebyshev polynomials of the first kind, that is,

$$T_n = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & & & & \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & & & \\ & \frac{1}{2} & \ddots & \ddots & & \\ & & \ddots & \ddots & \frac{1}{2} & \\ & & & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ & & & & \frac{1}{\sqrt{2}} & 0 \end{bmatrix}.$$

These matrices arise when one wants to compute the roots of a polynomial expressed in the first kind Chebyshev polynomial basis.

TABLE 6.1

*Errors for Type-*I *matrices of size* $n = 128$*, and* $\mathbf{u} = \alpha\,[1, \ldots, 1]^T$*. For all values of* α *our algorithm requires a time ranging from* 0.0070 *sec to* 0.0080 *sec, while the time required by the Lapack routine* `DGEEV` *ranges from* 0.0220 *to* 0.0260 *seconds.*

α	1	10^3	10^5	10^7	10^8	10^{11}
$E^{(abs)}$	1.0991e-14	9.0949e-13	2.9104e-11	7.4506e-09	1.4901e-08	4.7417e-08
$E^{(rel)}$	5.8831e-15	1.2950e-13	1.7515e-12	1.1038e-09	8.3495e-09	2.5190e-06
$E^{(rel2)}$	1.6396e-13	9.0885e-16	2.9104e-16	7.4506e-16	1.4901e-16	4.7417e-19
avrgit	2.5391	2.6328	2.7031	2.6484	2.8203	2.7500

TABLE 6.2

*Errors for Type-*I *matrices of the form* $T_n + \mathbf{u}\mathbf{e}_\mathbf{n}^\mathbf{T}$*, with* $\mathbf{u} = rand(n, 1)$*. For different values of* n*, the absolute and relative errors are shown. In the last two columns the times in seconds required by our method (*`GivQR`*) and by the Lapack routine* `DGEEV` *are given.*

n	$E^{(abs)}$	$E^{(rel)}$	$E^{(rel2)}$	`GivQR`	`DGEEV`
50	3.9968e-15	8.3890e-15	2.1393e-15	0.0010	0.0020
100	6.3283e-15	1.7494e-14	3.3873e-14	0.0050	0.0160
200	1.5543e-14	7.9124e-14	8.3196e-15	0.0160	0.0890
300	1.4655e-14	7.9692e-14	7.8442e-15	0.0360	0.2380
400	2.1094e-14	4.0091e-13	1.1291e-14	0.0600	0.4680
500	2.9310e-14	8.9860e-13	1.5688e-14	0.0920	0.7990
1000	2.9865e-14	1.3214e-12	1.5985e-14	0.3340	2.8410
2000	3.2707e-13	2.4688e-11	1.7507e-13	1.3030	11.8160
3000	8.1790e-13	6.1995e-11	4.3779e-13	2.9090	37.2560
4000	8.0880e-13	1.2163e-10	4.3292e-13	5.1890	70.2290

Type II: *Unsymmetric tridiagonal matrices.* We consider an almost symmetric tridiagonal matrix of the form

$$T = \begin{bmatrix} 0 & 1 & & & \\ 1 & 0 & \ddots & & \\ & \ddots & \ddots & 1 & \\ & & 1 & 0 & 1 \\ & & & \alpha & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & & & \\ 1 & 0 & \ddots & & \\ & \ddots & \ddots & 1 & \\ & & 1 & 0 & \alpha \\ & & & \alpha & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1-\alpha \\ 0 \end{bmatrix} \mathbf{e_n}^T. \tag{6.2}$$

When α becomes very large, the dominant eigenvalues are ill-conditioned.

Type III: *Hessenberg form of random matrices.* We consider the Hessenberg form of random symmetric plus random rank-one matrices.

In Table 6.1 we report the results for Type-I matrices, choosing $\mathbf{u} = \alpha\,[1, \ldots, 1]^T$ for increasing values of α. We note that the accuracy of the computed eigenvalues deteriorates for high values of α. For Type-I matrices with randomly chosen vector $\mathbf{u}$, we are able to obtain a higher accuracy, as reported in Table 6.2. Comparing with [7], we see that the results are comparable.

In Figure 6.2 the last two columns of Table 6.2 are plotted, and the data were fitted, respectively, with a quadratic and a cubic polynomial. This plot shows that, indeed, the quadratic behavior of our method can be observed from the beginning, and, moreover, it is always faster than the Lapack routine `DGEEV`.

We note that the accuracy in the computation of the eigenvalues deteriorates as α increases, suggesting a dependence on the norm of the perturbation. Comparing the results with those from [7], we see that a slight improvement in absolute value is made: at least one digit is gained.

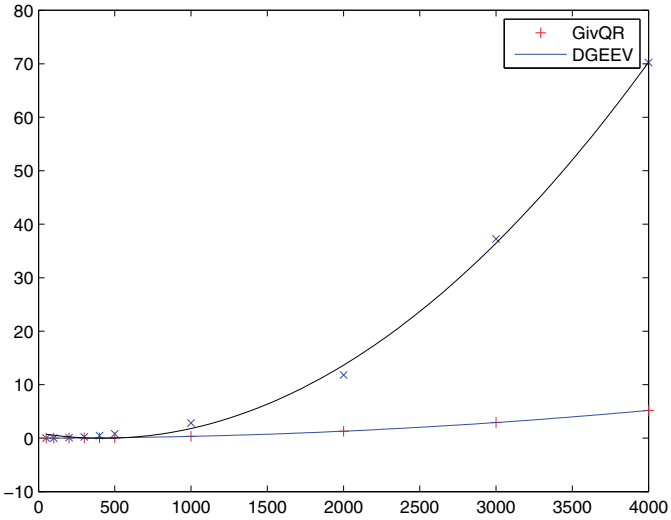


FIG. 6.2. *On the x-axis the size of the matrix. On the y-axis the time in seconds. The time required by our method (*`GivQR`*) increases quadratically with n, while the time required by* `DGEEV` *grows cubically. The data used are those reported in Table* 6.2*. The values obtained with* `GivQR` *are represented with crosses and the values obtained using* `DGEEV` *with asterisks.*

TABLE 6.3
*Errors for Type-*II *matrices, $n = 128$, and different values of α.*

α	1	10^2	10^4	10^6	10^8	10^{10}
$E^{(abs)}$	7.8826e-15	3.9080e-14	7.1054e-14	7.0145e-11	2.4975e-09	4.8511e-08
$E^{(rel)}$	2.1569e-13	1.5473e-13	3.4055e-13	7.0145e-11	2.4975e-09	4.4442e-07
$E^{(rel2)}$	3.9425e-15	3.8884e-15	7.1051e-16	7.0145e-14	2.4975e-13	4.8511e-13
avrgit	2.4922	2.7109	2.6797	2.7188	2.6406	2.5703

TABLE 6.4
Type III*: Hessenberg form of randomly generated symmetric plus rank-one matrices.*

n	$E^{(abs)}$	$E^{(rel)}$	$E^{(rel2)}$
50	5.5067e-14	5.5067e-14	8.6938e-16
100	3.1264e-13	2.9843e-13	1.9313e-15
200	3.0553e-13	1.8652e-13	1.2322e-15
500	5.4214e-12	9.8766e-13	8.4880e-15
1000	4.5906e-12	4.5906e-12	3.6693e-15

Matrices of Type II are almost symmetric tridiagonal matrices that can be expressed as a rank-one perturbation of a symmetric tridiagonal matrix. Depending on the magnitude of the perturbation, the extreme eigenvalues become ill-conditioned. Our algorithm is, however, very good at dealing with this situation (see Table 6.3) because for $\alpha = 10^{10}$ we still have an absolute error of $\mathcal{O}(10^{-8})$ and a relative accuracy of $\mathcal{O}(10^{-13})$, with respect to the infinity norm. Comparing with the results provided in [7], we see that for large values of α we gained significantly in accuracy.

Results for the Hessenberg form of randomly generated symmetric matrices plus random rank-one matrices are given in Table 6.4. We can see a slight loss of accuracy for higher values of n, which is due to the reduction to Hessenberg form, but this is still acceptable when looking at the condition numbers of the individual eigenvalues.

7. Conclusions. In this article a new algorithm was developed for performing a QR-step on a Hessenberg matrix, which is the sum of Hermitian plus low rank matrices. The implementation was based on the Givens-weight representation. The use of a unitary factorization to represent the matrix results in the incapability of using the standard deflation approach, and hence an alternative deflation technique was developed.

Numerical experiments illustrate that this algorithm is a viable alternative to the existing technique proposed in [7]. In some experiments a significant increase in accuracy is obtained, whereas in others the technique from [7] performs better. Comparing our implementation with Lapack routine `DGEEV`, we have that our method is already faster for $n = 50$.

Future research involves a backward error analysis and a detailed investigation of the deflation criterion used. Moreover, explicitly computing the last Givens transformation when performing the QR-iteration seems an unnatural step; it would be interesting if we could get rid of this explicit computation.

REFERENCES

[1] S. Barnett, *Polynomials and Linear Control Systems*, Marcel Dekker, New York, 1983.
[2] D. A. Bini, L. Gemignani, and V. Y. Pan, *Fast and stable QR eiegenvalue algorithms for generalized companion matrices and secular equations*, Numer. Math., 100 (2005), pp. 373–408.
[3] S. Chandrasekaran and M. Gu, *Fast and stable eigendecomposition of symmetric banded plus semi-separable matrices*, Linear Algebra Appl., 313 (2000), pp. 107–114.
[4] S. Delvaux and M. Van Barel, *Structures preserved by the QR-algorithm*, J. Comput. Appl. Math., 187 (2006), pp. 29–40.
[5] S. Delvaux and M. Van Barel, *A Givens-weight representation for rank structured matrices*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 1147–1170.
[6] S. Delvaux and M. Van Barel, *A Hessenberg reduction algorithm for rank structured matrices*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 895–926.
[7] Y. Eidelman, L. Gemignani, and I. C. Gohberg, *Efficient eigenvalue computation for quasiseparable Hermitian matrices under low rank perturbation*, Numer. Algorithms, 47 (2008), pp. 253–273.
[8] Y. Eidelman, L. Gemignani, and I. C. Gohberg, *On the fast reduction of a quasiseparable matrix to Hessenberg and tridiagonal forms*, Linear Algebra Appl., 420 (2007), pp. 86–101.
[9] D. Fasino, N. Mastronardi, and M. Van Barel, *Fast and stable algorithms for reducing diagonal plus semiseparable matrices to tridiagonal and bidiagonal form*, in Fast Algorithms for Structured Matrices: Theory and Applications, Contemp. Math. 323, AMS, Providence, RI, 2003, pp. 105–118.
[10] J. G. F. Francis, *The QR transformation: A unitary analogue to the LR transformation.* I, Comput. J., 4 (1961), pp. 265–271.
[11] J. G. F. Francis, *The QR transformation.* II, Comput. J., 4 (1962), pp. 332–345.
[12] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[13] N. Mastronardi, S. Chandrasekaran, and S. Van Huffel, *Fast and stable reduction of diagonal plus semi-separable matrices to tridiagonal and bidiagonal form*, BIT, 41 (2003), pp. 149–157.
[14] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
[15] R. Vandebril and Z. Bai, *An Implicit QR-Algorithm for Tridiagonal Plus Rank* 1 *Matrices*, Tech. report, Katholieke Universiteit Leuven, Leuven (Heverlee), Belgium, 2008.

[16] R. Vandebril, M. Van Barel, and N. Mastronardi, *A Note on the Recursive Calculation of Dominant Singular Subspaces*, Tech. report TW393, Department of Computer Science, Katholieke Universiteit Leuven, Leuven (Heverlee), Belgium, 2003.
[17] R. Vandebril, M. Van Barel, and N. Mastronardi, *A note on the representation and definition of semiseparable matrices*, Numer. Linear Algebra Appl., 12 (2005), pp. 839–858.
[18] R. Vandebril, M. Van Barel, and N. Mastronardi, *Matrix Computations and Semiseparable Matrices, Volume* I: *Linear Systems*, The Johns Hopkins University Press, Baltimore, MD, 2008.
[19] R. Vandebril, M. Van Barel, and N. Mastronardi, *Matrix Computations and Semiseparable Matrices, Volume* II: *Eigenvalue and Singular Value Methods*, The Johns Hopkins University Press, Baltimore, MD, 2008.
[20] R. Vanderbril and G. M. Del Corso, *An Implicit Multishift QR-Algorithm for Hermitian Plus Low Rank Matrices*, Tech. report TR-10-06, Department of Computer Science, University of Pisa, Pisa, Italy, 2010.
[21] D. S. Watkins, *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*, SIAM, Philadelphia, 2007.
[22] D. S. Watkins, *The QR algorithm revisited*, SIAM Rev., 50 (2008), pp. 133–145.
[23] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Numer. Math. Sci. Comput., Oxford University Press, New York, 1999.